Management of time-shifted IPTV services through transparent proxy deployment

T. Wauters, W. Van de Meerssche, F. De Turck, B. Dhoedt, P. Demeester
Dept. Information Technology (INTEC), Ghent University – IMEC – IBBT, Gaston Crommenlaan 8, bus 201
B-9050 Ghent, Belgium
tim.wauters@intec.ugent.be

T. Van Caenegem, E. Six
Alcatel R&I, Access and Edge
Francis Wellesplein 1
B-2018 Antwerp, Belgium

Abstract— **A recent important evolution in broadband access network design is the deployment of IP aware access network elements, which allow to introduce access network services beyond basic triple-play. The focus of this paper is on the management of time-shifted television (tsTV), an IPTV service which allows for watching the broadcast content at real-time or with a (small) time shift. An architecture for a large-scale tsTV service deployment is presented, using co-operating transparent diskless proxy caches in broadband access networks, with an implementation based on the IETF's Real-Time Streaming Protocol (RTSP). Caching algorithms have been designed to take into account content popularity and distance metrics. The algorithms make use of the sliding window concept and calculate the optimal trade-off between bandwidth usage efficiency and storage cost. A prototype implementation of a transparent tsTV proxy is presented and evaluated through performance measurements.**

Keywords- IPTV; time-shifted TV; real-time algorithms; transparent proxy; RTSP

I. INTRODUCTION

During the last few years, the architectural model of access networks has evolved towards multi-service and multi-provider networks. Ethernet as well as full IP alternatives have been investigated as viable connectionless successors for the legacy ATM-based platforms. While the introduction of Ethernet up to the edge (e.g. through VLANs) solves some the of the existing PPP problems, new ones are created. Traffic segregation issues because of address resolution complications prevent large-scale access network deployments and therefore an IP-aware network model [8] is often considered a valuable alternative.

One of the emerging services is television over IP. Currently, IPTV services are generally limited to Video on Demand (VoD) and Broadcast TV. VoD servers are typically located at the edge of the core network and put a heavy burden on the access networks in case of large deployments, possibly causing the network to congest. The solution proposed in this paper is to focus on time-shifted television (tsTV) and to deploy distributed caches and streamers in the aggregation nodes, co-operating on both a peer-to-peer and a hierarchical level. This approach offers an alternative for deploying dedicated home equipment for video storage, such as a home Personal Video Recorder (PVR), that has limited throughput capacity and is rather expensive.

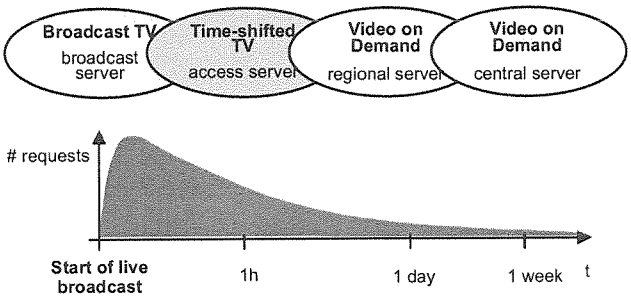


Figure 1. Delivery mechanisms for IPTV

Time-shifted TV enables the end-user to watch a broadcasted TV program with a time shift, i.e. the end-user can start watching the TV program from the beginning although the broadcasting of that program has already started or is even already finished. As shown in Fig. 1, the popularity of a live television program typically reaches its peak within several minutes after the initial broadcast of the program and decreases roughly exponentially afterwards. This means that caching a segment with a sliding window of several minutes for each current program can serve a considerable part of all user requests for that program, from start to finish, hence the benefit of using distributed streamers with limited storage capacity. In Fig. 2a and 2b for example, user 1 is the first to request a certain television program and gets served from the central server. Afterwards, other requesting users (e.g. user 2) can be served by the proxy, as long as the window of the requested program is still growing. After several minutes, the window stops growing and begins sliding, so that user 3 cannot be served anymore and will be redirected to the (central or regional) server or, in case of co-operative caching, to a neighbor proxy with the appropriate segment, if present. Pausing (parallel to the horizontal axis, Fig. 2b) can also be supported within the segment window, as well as fast forward or rewind (parallel to the vertical axis).

The remainder of this paper is structured as follows. Research work related to this study is briefly discussed in section 2.

This work is partly funded by the IST FP6 MUSE project. MUSE contributes to the strategic objective "Broadband for All" of IST (Information Society Technologies) and it is partially funded by the European Commission.

1-4244-0357-X/06/$20.00 ©2006 IEEE

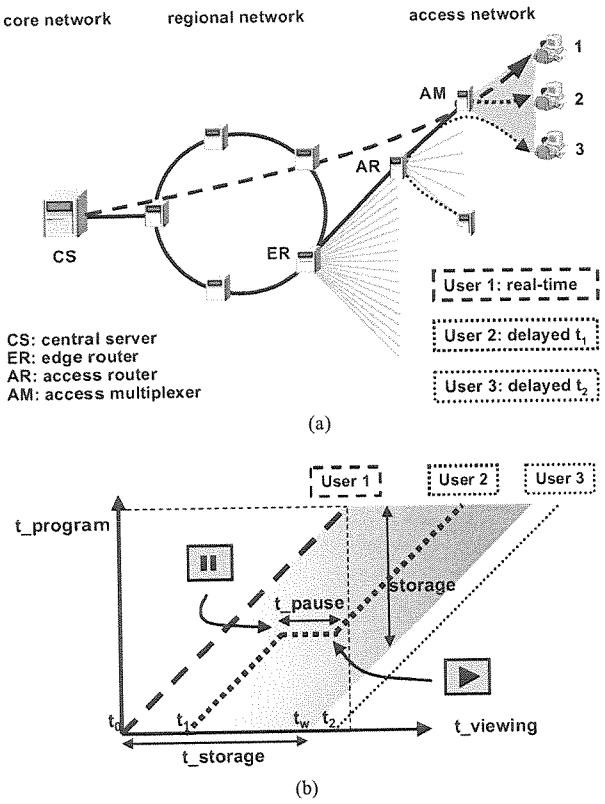


Figure 2. Time-shifted television: (a) typical access network topology and (b) tsTV streaming diagram

Section 3 presents our sliding-interval caching algorithm. It determines the location and the size of the different segments at the proxy caches. Different deployment options are discussed as well. In section 4, the RTSP implementation is detailed and evaluated through measurements. Section 5 concludes this paper.

II. BACKGROUND AND RELATED WORK

Previous studies on proxy caching techniques [1] or distributed replica placement strategies for content distribution networks (CDNs) [4] show that greedy algorithms that take distance metrics and content popularity into account perform better than more straightforward heuristics such as LRU (Least Recently Used) or LFU (Least Frequently Used). Segment-based caching techniques have been studied extensively for streaming media, due to the huge size of multimedia streams compared to traditional web objects. A survey on different segment-based strategies has been presented in [1]. Of particular interest for this study is sliding-interval caching [2], where the cached portion of the stream is initially a growing prefix, but afterwards changes into a dynamically updated sliding interval. This way, consecutive requests can be served from start to finish within this window. A more advanced aspect is the use of co-operative proxy caching [3], where a better performance and system scalability than with independent proxies can be achieved through load balancing. Several studies such as [7] have been investigating the implementation of segment-based caching techniques on proxies using the RTP / RTCP / RTSP protocol suite. A demonstrator of an IP aware multi-service access network, including our prototype tsTV setup, has been described in [8].

In this paper, we combine the benefits of sliding-interval caching and co-operative caching, and introduce a RTSP-based proxy implementation supporting our algorithms.

III. SLIDING-INTERVAL CACHING ALGORITHM

Our caching algorithm for tsTV services is presented in this section. Since we assume that in general only segments of programs will be stored, cache sizes can be limited to a few gigabyte or even less in case of co-operative caching. This way smaller streaming servers can be deployed closer to the users, without increasing the installation cost excessively.

A. Basic principles

We propose that the cache is virtually split up in two parts: a small part S and a main part L. Part S will be used to cache the first few (e.g. 5) minutes of every newly requested (or broadcasted) program, mainly to determine its initial popularity. Its size is generally smaller than 1 GB (typically 1 hour of streaming content).

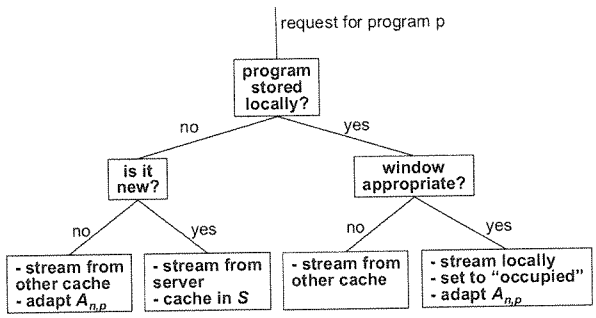


Figure 3. Basic principle of the tsTV caching algorithm at each proxy

Part L will be used to actually store the appropriate segments (with growing or sliding windows). This part is again virtually divided into two separate storage spaces. Part L_1 is used to store unique segments only, shared among all cache nodes, co-operating at the same access network level. We assume that all caches know which unique segments are stored on the other caches, through a Cache State Exchange (CSE) protocol. This way, all parts L_1 on all cache nodes represent one large cache, mainly to offload the central server. The second part L_2, if there is still storage space left, is then used to store segments that are locally most popular. The main goal of that part is to offload the access network links, used by the co-operative caching mechanism (requests served by L_1 on a neighbor cache). The actual size of each segment in part L_2 will be determined and, if necessary, adapted after each interval Δ (e.g. 5 minutes). Fig. 3 shows the basic principle of the tsTV caching algorithm. During each interval Δ, program requests arrive at the different proxies. Each time, a parameter $A_{n,p}$ will be updated in proxy n, for program p. In general, this parameter tries to determine the popularity of the program, while taking distance metrics into account. This means that a popular program might not be cached, because a nearby proxy already

1-4244-0357-X/06/$20.00 ©2006 IEEE

stores that program. If $A_{n,p}$ for program p at cache node n is high enough, the window will grow, otherwise the window will start sliding, or will be dropped in case no requests are currently being served within that segment. $A_{n,p}$ is calculated as follows:

Every time a request for program p arrives at proxy n, $A_{n,p}$ is increased by 1 (only taking popularity into account) or by the hopcount between proxy n and the serving node (taking popularity and distance into account).

After each interval Δ, first all segments (sliding or growing) with status set to "occupied" (i.e. serving requests) are stored in L_2. Afterwards L_2 is filled with segments with growing windows for the most popular programs (i.e. with the highest values of $A_{n,p}$). All other segments are dropped, S is cleared and all values of $A_{n,p}$ are reset to 0.

B. Deployment options

To demonstrate the different deployment options, simulations were performed on the typical access network topology shown in Fig. 2a. The first set demonstrates the stand-alone (hierarchical) mode, where we can consider that $L = L_2$. The second set introduces co-operative caching ($L = L_1 + L_2$).

1) Hierarchical caching: To illustrate the hierarchical caching principle, simulations were performed on one branch of the access network tree of Fig. 2a: a regional server at the edge node with two hierarchical caches (c1 at the access router and c2 at access multiplexer). The server offers 5 popular channels through the tsTV service, each with 6 programs of 45 minutes per evening (2.5 Mbps streams). The popularity of each program reaches a peak during the first interval (Δ = 5 minutes) after the start and is halved after each interval (similar to Fig. 1). Fig. 4 shows the load on the link between the server and cache c1 (s -> c1 (hierarchical)), where all requests for one program are made within 30 minutes after it has started. The central server load already drops to 50% when 0.5GB caches are used at level 1 and 2. What happens is that the caches c1 and c2 first store all 5-minute prefixes of each new program in S, but since only cache c2 intercepts new requests afterwards, cache c1 will drop these segments after Δ. Afterwards cache c1 will store the next 5 minutes of each program, since cache c2 is full and stores the sliding "occupied" windows from the first interval. This means that the caches serve all requests made during the first 10 minutes of each program.

These results are very similar to the results of our exact analytical model [11], based on fixed-sized intervals, that offers a method to estimate the required storage space in the network.

2) Co-operative caching: Contrary to hierarchical caching, where a request that cannot be served is forwarded to the next cache on the path to the server, caches can now forward requests to caches on the same level. Since all caches know the cache states of all other nodes, all parts L_1 can be effectively filled, while the rest of the storage space (L_2) can be used for locally popular content. The filling of L_1 can be optimized using the "transparent RTSP request forwarding" principle, explained in the following section, to effectively create one large virtual cache.

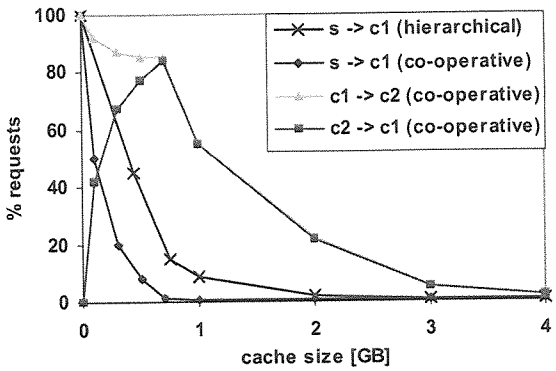


Figure 4. Fraction of the streams on the links between the server and the level 1 node (s -> c1) and between the level 1 and 2 nodes (downlink c1 -> c2 and uplink c2 -> c1), in hierarchical and co-operative mode (all requests made within 30 minutes)

This can also be observed in Fig. 4. The simulations are now performed on the access network topology of Fig. 2a, with 6 co-operating level 2 caches (at the access multiplexers) over bi-directional links, one access router and a regional server at the edge node. No level 1 cache (at the access router) is assumed, to avoid any influence from hierarchical caching. The cost of using the link from the regional server to the access router (s -> c1 (co-operative)) has been set to a value higher than 1, to avoid using the server when the requested segment can already be found on another level 2 cache (when calculating the shortest path using the weighted Dijkstra algorithm). The server load decreases six times faster to (almost) zero than in stand-alone mode, when only the part L_1 is used (<1GB per cache) on all six proxy caches. First the load on the uplinks in the access network increases due to cache co-operation and is fully balanced with the downstream traffic when the server load drops to (almost) zero. Afterwards, when the storage space per cache is large enough (>1GB), the load on the bi-directional access network links is reduced as well, since most requests can be served locally.

Hierarchical and co-operative caching are complementary techniques and might be combined to provide even better results for small caches sizes (<1GB).

IV. TSTV SERVICE DEPLOYMENT

A transparent RTSP proxy for time-shifted TV has been implemented (in C++) for evaluation purposes. This section gives an overview of the different components and protocols used and evaluates a prototype through performance measurements.

A. Functionality

In order to implement the proxy, its functionality is divided into logical parts. The communication with the users and the central server includes messages containing data about which program or channel has to be streamed, or VCR like commands such as PAUSE and STOP. A protocol commonly used for this interaction is RTSP (Real-Time Streaming Protocol) [5]. The streams themselves are encapsulated and delivered with RTP (Real-Time Protocol), a standard protocol for live streamed

1-4244-0357-X/06/$20.00 ©2006 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.

media [6]. A first functional component of the proxy is the *RTSP Proxy*, a component that communicates with the tsTV clients and the server using RTSP, interprets their messages and commands the other components to execute these requests. The *RTSP Proxy* component delegates the caching algorithm decisions to another component, the *Cache Verdict Manager*, a component that uses information from the *Cache State Manager*, which is updated through a centralized or distributed Cache State Exchange (CSE) protocol. The task of the *Cacher* component is to store popular streams, sent to the proxy by the server (or another cache), in sliding windows. The streams are sent to the clients from these windows, a function that is handled by the *Streamer* component. The proxy also keeps track of the streams that are being sent to the proxy (which program, channel, starting time, ...), through the *Stream Tracker* component, with help from the *Program Guide* component, which communicates with the electronic program guide (EPG) server. The *Packet Handler* acts as an interface, dealing with low-level network interaction. Fig. 5 gives an overview of the different components.

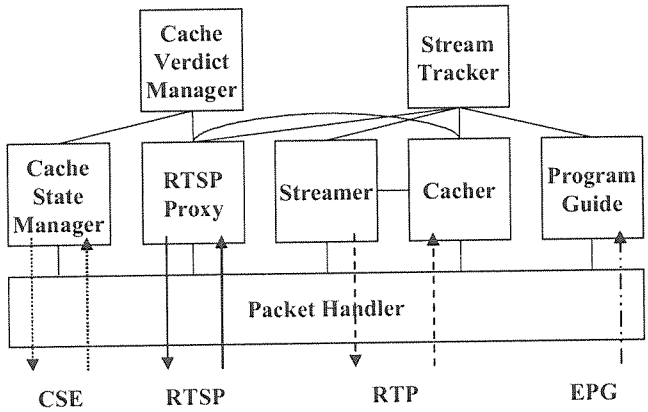


Figure 5. Overview of the different components in the proxy cache

B. *Detailed scenario*

Fig. 6 shows a detailed setup of a streaming session between the client, the proxy caches and the server. First, the client sends an RTSP request to the server, but this request is intercepted by the proxy. In a first scenario (Fig. 6, 1a), the proxy does not store the requested fragment, forwards the request (with the destination IP address of the proxy) to the server, starts caching the stream from the server and forwards the RTP stream to the user. Afterwards the proxy exchanges its new cache state in a distributed way to all other caches (Fig. 6, 2a). In a second possible scenario (Fig. 6, 1b), the proxy does not store the requested fragment and decides not to store the fragment locally. It forwards the RTSP request to another (proxy) cache, keeping the destination IP address of the client. The other proxy decides to forward the request to the server, caches the fragment locally and sends the RTP stream directly to the client. Afterwards, the new cache states are exchanged through a centralized CSE protocol (Fig. 6, 2b). The second scenario shows how the co-operative caching algorithm (section III.B.2) can efficiently create one large virtual cache, using the "transparent RTSP request forwarding" principle.

C. *Test setup and measurements*

In this section, performance measurements on a prototype proxy are presented, implemented on an AMD AthlonTM 64 processor 3000+ (512MB RAM). Fig. 7 shows the number of client RTSP requests that can be handled simultaneously by the proxy, already serving RTP streams (2.5Mbps) over a gigabit link (560Mbps throughput measured with Iperf [9]). The proxy uses high-priority RTP threads and low priority RTSP threads. We observe that the RTSP handling decreases linearly and fails at 190 simultaneous RTP streams (480Mbps), due to limited system resources. Fig. 8 shows the delay between a PLAY request sent by a PC client and the arrival of the first RTP packet at the PC client, for different configurations (server-proxy-client). Even when the proxy has to fetch the content from the server, the delay is never higher than 35 ms (1000 measurements per configuration). When the proxy acts as a mere router, the delay caused by the server (Darwin streamer [10]) is less than 1 ms. The delay on the network links between server, proxy and client is negligible.

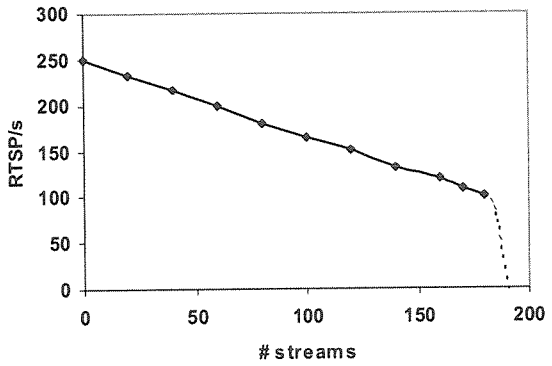


Figure 6. RTSP requests handling (AMD AthlonTM 64 processor)

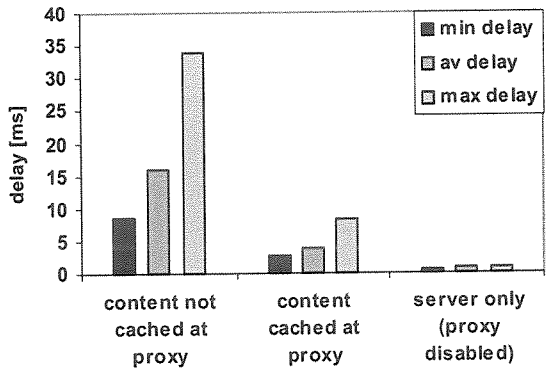


Figure 7. Delay between a client request and the actual start of the RTP stream on a client PC

V. CONCLUSIONS

In this paper a novel architecture as well as caching algorithms for time-shifted television are presented. Cache decisions at low cost distributed streamers storing sliding intervals are made after fixed learning intervals, based on popularity and distance metrics.

1-4244-0357-X/06/$20.00 ©2006 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.

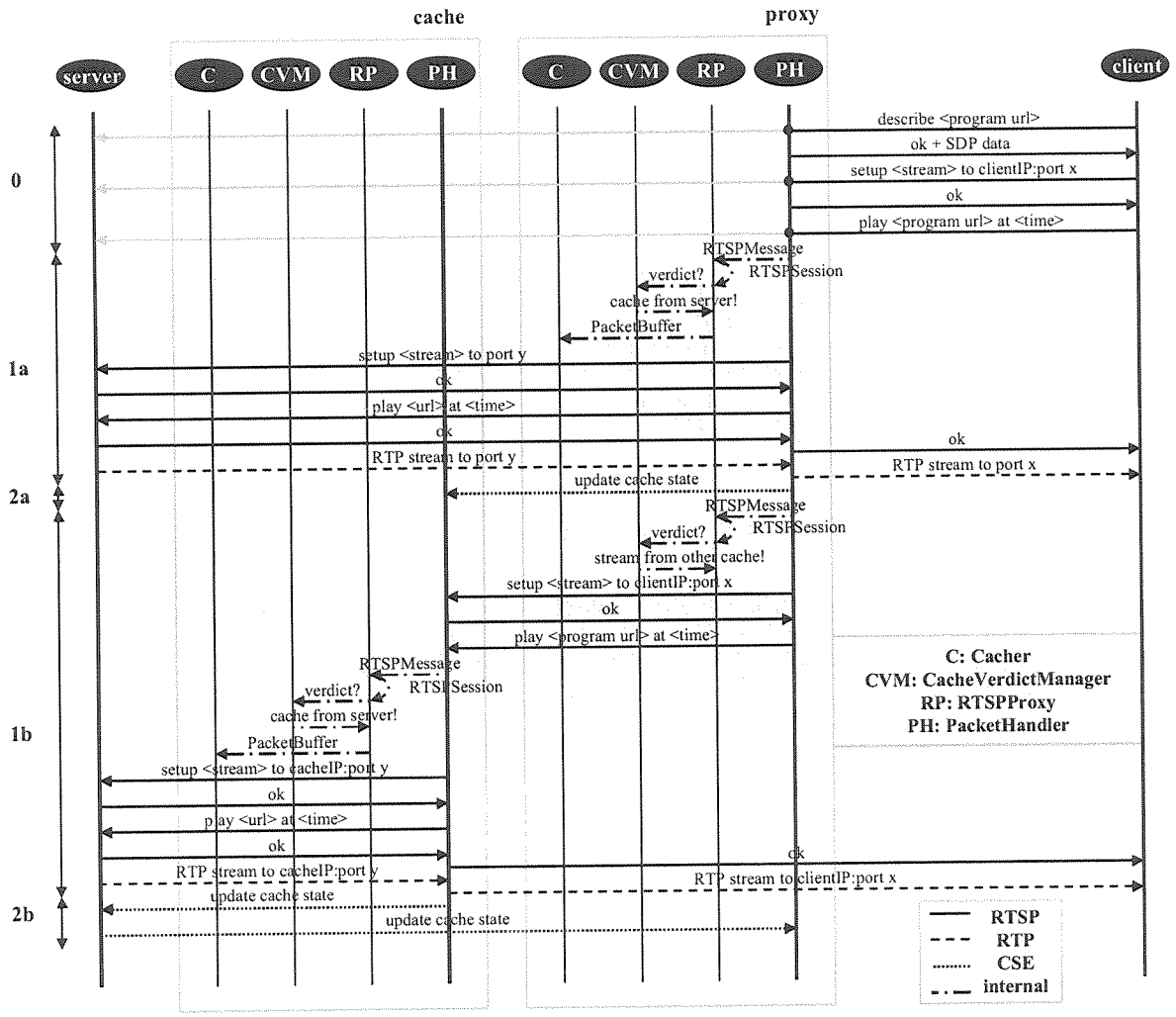


Figure 8. Detailed setup of a streaming session between client, proxy, any other cache and the server. In scenario a, the proxy caches the requested program from the server; in scenario b, the proxy forwards the RTSP request transparently to another cache

Experimental results for a typical access network topology show promising results in terms of server and network load, especially for co-operative caching. An transparent RTSP proxy implementation has been introduced as well. A prototype integrating the caching algorithms has been built and evaluated through measurements.

REFERENCES

[1] J. Liu, J. Xu, "Proxy caching for media streaming over the Internet", IEEE Communications Magazine, vol. 42, no. 8, August 2004, pp. 88-94.

[2] S. Chen, B. Shen, Y. Yan, S. Basu, X. Zhang, "SRB: Shared running buffers in proxy to exploit memory locality of multiple streaming media sessions", in proceedings of the 24th IEEE International Conference on Distributed Computing Systems (ICDCS), 2004.

[3] Y. Chae, K. Guo, M. Buddhikot, S. Suri, E. Zegura, "Silo, rainbow, and caching token: schemes for scalable, fault tolerant stream caching", IEEE Journal on Selected Areas in Communications, vol. 20, no. 7, September 2002, pp. 1328-1344.

[4] M. Karlsson, C. Karamanolis, M. Mahalingam, "A framework for evaluating replica placement algorithms", Technical Report HPL-2002, HP Laboratories, July 2002.

[5] RFC 2326, Real Time Streaming Protocol (RTSP).

[6] RFC 1889, RTP: A Transport Protocol for Real-Time Applications.

[7] S. Gruber, J. Rexford, A. Basso, "Protocol considerations for a prefix-caching proxy for multimedia streams", Computer Networks 33 (1-6) (2000) 657-668.

[8] E. Gilon, W. Van de Meerssche et al., "Demonstration of an IP aware multi-service access network", in proceedings of BroadBand Europe 2005, December 2005, Bordeaux, France.

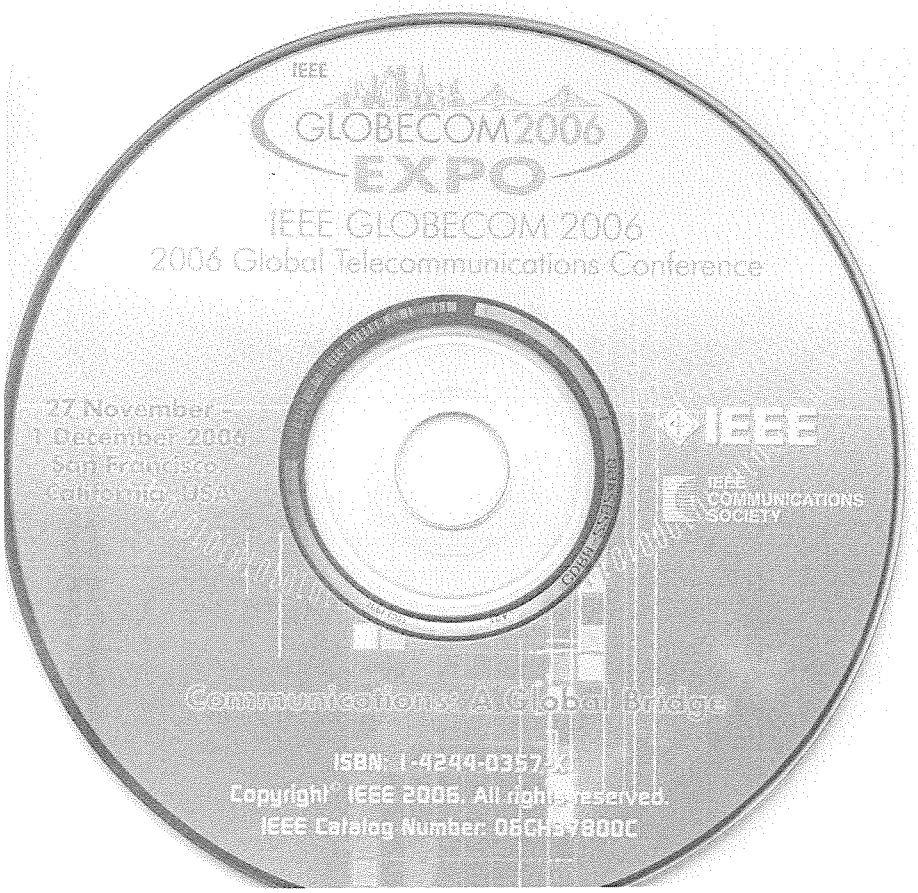
[9] Iperf, TCP/UDP bandwidth measurement tool, http://dast.nlanr.net/projects/Iperf

[10] Darwin Streaming Server, http://developer.apple.com/opensource/server/streaming

[11] T. Wauters et al., "Co-operative Proxy Caching Algorithms for Time-Shifted IPTV Services", in proceedings of EuroMicro 2006, September 2006, Dubrovnik, Croatia.

1-4244-0357-X/06/$20.00 ©2006 IEEE

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.



ISBN: 1-4244-0357-X

IEEE Catalog Number: 06CH37800C